# Do README ao AI Manifest: O Próximo Padrão dos Projetos de Software

*Definição da especificação do AI Manifest (.ai/manifest.md), um contrato declarativo de baixo acoplamento projetado para descrever a stack, arquitetura e limites operacionais de um repositório para agentes de IA.*

**Autor:** Maicon Santos Webmaicon
**Advisor:** Lunara LU-WANING 260710

**Criado em:** 10/06/2026
**Atualizado em:** 13/06/2026

#AIManifest #SoftwareEngineering #AgenticAI #Documentation #Standardization #Repository #SecurityFirewall #IDE #Workflow #ContextAsCode

---

## Do README ao AI Manifest: O Próximo Padrão dos Projetos de Software

### Resumo Executivo

Por décadas, os projetos de software foram estruturados sob a perspectiva de leitores humanos e ferramentas de automação tradicionais. Arquivos como `README.md`, `package.json`, `Dockerfile` e variáveis de ambiente `.env` definem como configurar, compilar, testar e distribuir um sistema.

Contudo, com a ascensão dos agentes autônomos de Inteligência Artificial integrados ao fluxo de desenvolvimento diário, surge uma nova necessidade metodológica: **os repositórios de código precisam se autodescrever de forma operacional para IAs.**

Embora o `README.md` continue indispensável para a integração de humanos, ele falha ao fornecer as restrições operacionais e os limites de atuação exigidos por um agente autônomo. O `AI Manifest` surge para preencher este gap: ele funciona como um contrato operacional explícito que dita arquitetura, políticas de permissão, stack tecnológica, padrões de qualidade e limites de segurança que governam as ações de agentes de IA dentro do ecossistema do projeto.

---

### Premissas Fundamentais

> 💜 **IMPORTANT**
> A eficácia operacional de uma IA em um repositório é inversamente proporcional à sua necessidade de inferir ou improvisar.

A especificação do AI Manifest baseia-se em sete princípios básicos: 1. **Natureza Declarativa:** A documentação para humanos é predominantemente narrativa; o contexto para agentes de IA deve ser declarativo, resumido e estruturado. 2. **Precedência da Arquitetura:** O agente não deve inferir ou reinventar padrões arquiteturais; ele deve aderir estritamente às decisões documentadas do repositório. 3. **Firewall Operacional:** As permissões de alteração, leitura e execução de comandos pelo agente devem ser explícitas, determinísticas e restritivas. 4. **Fonte Canônica da Verdade:** O repositório deve conter um único ponto de referência central de contexto, evitando a dispersão de instruções em múltiplos arquivos proprietários de fornecedores. 5. **Desacoplamento de IDEs:** Adapters de ferramentas específicas (ex. Claude, Cursor, Copilot) devem ser derivados do contexto canônico, e não poluidores da raiz do repositório. 6. **Versionamento Nativo:** O manifesto operacional deve evoluir e ser versionado em conjunto com o código-fonte (`Context as Code`). 7. **Eficiência Cognitiva:** O contexto do projeto precisa ser modular, leve e de fácil ingestão para reduzir o consumo de tokens e evitar alucinações.

---

## O Limite do README: O Gap do Desenvolvimento Assistido

O arquivo `README.md` foi projetado para responder ao *"como entender"* e *"como começar"* de um desenvolvedor humano. Ele explica o propósito macro do sistema e ensina a instalar dependências.

No entanto, um agente de IA executa ações de modificação direta no código. Para operar com segurança e precisão, ele necessita de respostas para perguntas que o `README.md` normalmente omite: * Quais padrões de nomenclatura exatos devem ser aplicados para novas classes ou componentes? * Quais diretórios e arquivos são estritamente sensíveis (firewall lógico)? * Quais comandos de sistema são seguros para execução e quais exigem aprovação humana? * Quais decisões arquiteturais passadas delimitam a implementação atual? * Como fragmentar tarefas complexas sem violar a consistência do projeto?

Sem essas diretrizes declarativas, a IA opera sob um viés probabilístico genérico, o que leva a improvisações e, consequentemente, a regressões ou bugs de arquitetura.

---

## A Proposta do AI Manifest

O **AI Manifest** centraliza e organiza todo o contexto operacional de IA em um diretório específico. A recomendação central de design é utilizar a pasta `.ai/` na raiz do projeto devido à sua semântica limpa e facilidade de varredura por assistentes.

```
.ai/
├── manifest.md            # Contrato principal e indexador de regras
├── core/                  # Fonte canônica e regras de governança locais
│   ├── project.md         # Escopo de negócio e vocabulário do domínio
│   ├── stack.md           # Tecnologias oficiais e versões aceitas
│   ├── architecture.md    # Padrões de software e organização de pastas
│   ├── permissions.md     # O Firewall Operacional (políticas de acesso)
│   ├── guidelines.md      # Padrões de código e convenções estilísticas
│   ├── testing.md         # Estratégia de testes e validação
│   ├── git.md             # Padrões de ramificação, commits e Pull Requests
│   ├── commands.md        # Lista de comandos autorizados e restritos
│   └── risks.md           # Pontos de atenção e zonas críticas do sistema
├── shared/                # Capacidades e fluxos reutilizáveis
│   ├── skills/            # Procedimentos passo a passo (capacidades da IA)
│   ├── agents/            # Definições de papéis e personas de agentes
│   ├── workflows/         # Rotinas e automações sequenciais
│   └── rules/             # Regras compartilhadas entre múltiplos projetos
├── vendor/                # Dependências externas de contexto (somente leitura)
├── adapters/              # Configurações de tradução para IDEs específicas
│   ├── claude/            # Configurações para Claude Code / CLAUDE.md
│   ├── cursor/            # Configurações para Cursor (.cursorrules)
│   ├── copilot/           # Configurações para GitHub Copilot
│   └── windsurf/          # Configurações para Windsurf
├── decisions/             # Registro de Decisões Arquiteturais (ADRs)
│   ├── 0001-auth.md
│   └── 0002-billing.md
└── changelog.md           # Histórico de alterações operacionais efetuadas por IAs
```

---

## Detalhamento da Estrutura Core

### 1. `manifest.md`

É a regra zero. Funciona como ponto de entrada e contrato de inicialização da IA.

```
# AI Manifest

Este repositório autoriza assistência por agentes de IA, sob a condição de aderência estrita às reg

Antes de propor ou modificar qualquer arquivo, o agente deve ler:
- `.ai/core/permissions.md` (prioridade máxima)
- `.ai/core/project.md`
- `.ai/core/architecture.md`

Em caso de ambiguidades de instrução, a política em `permissions.md` prevalece sobre qualquer outra
```

### 2. `project.md`

Descreve o domínio de negócio de forma concisa para evitar que a IA trate a aplicação como um CRUD genérico.

```
# Escopo do Projeto

Este sistema gerencia campanhas publicitárias conectando marcas a influenciadores digitais.

## Termos de Domínio Fundamentais:
* **Match Score:** Algoritmo que calcula afinidade entre marca e criador.
* **Lockout Period:** Período obrigatório de não concorrência pós-campanha.
* **Flag de Risco:** Marcação automática de potenciais conflitos éticos.
```

### 3. `stack.md`

Define o ecossistema tecnológico formal do projeto. Impede que a IA sugira ou injete dependências inadequadas.

```
# Stack Tecnológica Oficial

## Backend:
* Laravel v11.x (PHP 8.3)
* Pest (Framework de Testes)
* PostgreSQL v16 (Banco de dados principal)

## Frontend:
* Livewire v3.x & Alpine.js
* Tailwind CSS v3.x
```

### 4. `architecture.md`

Delimita as camadas do sistema e a separação de responsabilidades.

```
# Diretrizes Arquiteturais

* **Controllers:** Apenas orquestram chamadas; sem regras de negócio ou queries SQL.
* **Services:** Concentram a lógica de negócio e regras do domínio.
* **Form Requests:** Responsáveis únicos por validação de entrada de dados.
* **Events/Listeners:** Utilizados para desacoplar ações secundárias (ex.: envio de e-mails).
```

### 5. `permissions.md` (O Firewall Operacional)

Garante a segurança da operação limitando o raio de ação das IAs. É o componente mais importante do manifesto.

```
# Políticas de Acesso e Escrita (Firewall)

## Zonas Livres (Sem necessidade de validação adicional)
* `app/Services/`
* `app/Http/Controllers/`
* `tests/`

## Zonas Restritas (Exigem confirmação explícita do desenvolvedor)
* Criação ou alteração de migrations de banco de dados.
* Fluxos críticos: Autenticação, Faturamento (Billing), e Permissões (ACL).

## Zonas Proibidas (Bloqueio total de alteração direta)
* Arquivos de configuração sensitiva (`.env`, secrets, credenciais).
* Chaves criptográficas e certificados.
```

### 6. `guidelines.md`

Instrui sobre qualidade de código e convenções estilísticas recomendadas para o projeto.

```
# Diretrizes de Código

* **Clean Code:** Priorize nomes de variáveis e métodos descritivos antes de brevidade.
* **Legibilidade:** Código limpo e autodocumentado é preferível a comentários redundantes.
* **Padrões:** PHP em conformidade estrita com PSR-12.
```

---

## Modularidade e Reuso: Skills e Workflows

Para acelerar tarefas comuns sem redundância de instrução no prompt do usuário, o manifesto organiza rotinas operacionais em `skills/` e `workflows.md`.

### Skills (Habilidades Procedimentais)

São guias passo a passo de como executar uma tarefa técnica repetitiva.

```
# Skill: Criar Endpoint de API

1. Analise o modelo de dados afetado.
2. Crie um Form Request específico para validação em `app/Http/Requests`.
3. Implemente as regras de negócio em um Service exclusivo.
4. Exponha o endpoint via Controller, retornando uma Resource de resposta.
5. Escreva um teste de integração (Feature test) cobrindo caminhos felizes e de erro.
```

### Workflows (Fluxos de Processo)

Definem rituais operacionais do ciclo de desenvolvimento.

```
# Workflow: Nova Feature

1. Leia o escopo da tarefa e consulte `project.md`.
2. Identifique os módulos afetados e busque referências em `architecture.md`.
3. Desenvolva a funcionalidade acompanhada de seus testes.
4. Execute a suite de testes locais.
5. Registre a mudança em `changelog.md` seguindo a semântica do projeto.
```

---

## Resolução de Conflitos e Precedência

Em cenários complexos onde regras do time, da empresa ou da IA colidem, a árvore de precedência é executada de forma determinística de cima para baixo:

```
1. permissions.md (Regra de segurança local do repositório)
   └── 2. architecture.md (Padrão arquitetural do repositório)
       └── 3. decisions/ (Decisões arquiteturais históricas locais)
           └── 4. stack.md (Stack tecnológica local)
               └── 5. guidelines.md (Padrões de estilo locais)
                   └── 6. Regras Corporativas (Diretrizes globais da empresa)
                       └── 7. Preferências Pessoais (Configurações do desenvolvedor)
```

---

## Abstração por IDEs: A Camada de Adapters

Provedores de IA utilizam nomenclaturas proprietárias na raiz do repositório. O padrão resolve isso isolando o contexto canônico na pasta `.ai/` e utilizando geradores ou scripts de tradução para mapear estes arquivos para os destinos corretos:

- **Claude Code →** Lê de `.ai/core` ou gera `CLAUDE.md`.
- **Cursor →** Mapeia para `.cursorrules` ou arquivos na pasta `.cursor/`.
- **GitHub Copilot →** Mapeia para `.github/copilot-instructions.md`.
- **Windsurf →** Mapeia para `.windsurf/rules/`.

Desta forma, os desenvolvedores atualizam apenas a documentação canônica em `.ai/core/` e executam um utilitário para propagar o contexto atualizado para todas as IDEs suportadas.

---

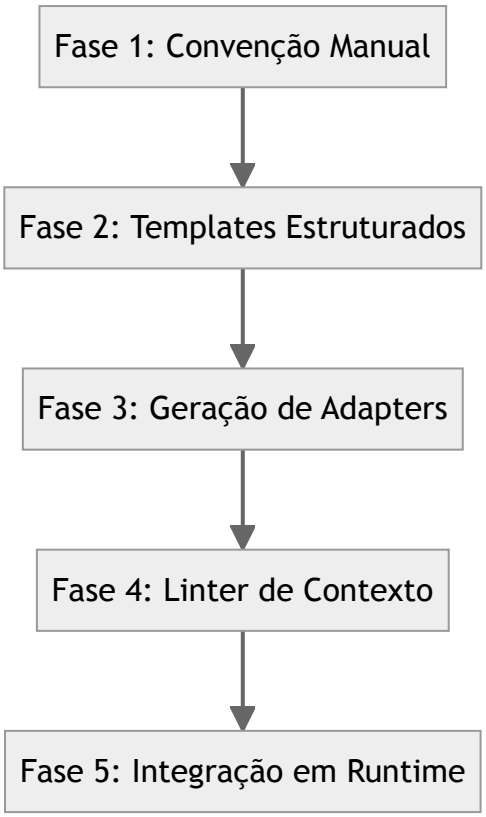
## Análise de Riscos e Mitigações

- **Risco de Obsolescência:** Documentos de contexto desatualizados induzem a IA a introduzir padrões legados no código.
- *Mitigação:* Valide a consistência das regras no pipeline de Integração Contínua (CI), verificando se modificações arquiteturais foram acompanhadas de atualizações na pasta `.ai/`.
- **Excesso de Especificação:** A imposição de regras excessivamente rígidas limita a capacidade de resolução de problemas do agente de IA.
- *Mitigação:* O manifesto deve focar em *limites e restrições*, não em microgerenciamento de algoritmos.
- **Injeção de Contexto Malicioso:** Pacotes em `vendor/` importados de fontes não confiáveis podem conter prompts maliciosos projetados para exfiltrar segredos.

- *Mitigação:* Realize revisões de segurança e assine digitalmente dependências de contexto corporativo.

---

## Roadmap Conceitual de Adoção




- **Fase 1 — Convenção Manual:** Implementação direta do diretório `.ai/` e seus arquivos fundamentais no repositório.
- **Fase 2 — Templates Estruturados:** Criação de templates de manifesto pré-configurados por stack (ex. Laravel, Node, React).
- **Fase 3 — Geração de Adapters:** Automatização do mapeamento do contexto canônico para os padrões das IDEs (Cursor, Windsurf, Claude).
- **Fase 4 — Linter de Contexto:** Ferramenta para analisar a consistência das regras e garantir que não haja instruções contraditórias.
- **Fase 5 — Integração em Runtime:** Consumo automatizado e dinâmico das diretivas diretamente pelos ambientes de execução dos agentes.

---

## Referências Técnicas de Contexto

Estas referências servem como fundação técnica e pontos de contato com as especificações vigentes no mercado:

- **Model Context Protocol (MCP):** Especificação de Protocolo
- **MCP Roots:** Client Roots Specification
- **MCP Resources:** Server Resources Specification
- **Claude Code Docs:** Memória e uso de CLAUDE.md

- **GitHub Docs:** Instruções customizadas do Copilot
- **Apple Security Research:** Private Cloud Compute Architecture
- **Microsoft Learn:** Arquitetura Windows AI e NPU Devices

---

## Conclusão

O `README.md` foi projetado para um ecossistema onde humanos detinham o monopólio da escrita e alteração de código. No novo paradigma de engenharia assistida, o projeto exige uma camada de governança transparente e acionável para inteligências artificiais.

O `AI Manifest` não é apenas um guia estático de documentação. É um contrato de convivência e produtividade entre o engenheiro humano e o agente de inteligência artificial. No futuro do desenvolvimento de software, o diferencial de um projeto não será apenas a qualidade do código produzido, mas o quão bem estruturado ele está para que a IA colabore com máxima eficiência e mínima supervisão.


### Sobre o Autor

Maicon Santos (Webmaicon) é Tech Lead, AI Solutions Architect e desenvolvedor Full Stack com 27 anos de experiência em tecnologia. Pioneiro em segurança digital (tendo criado um sistema de 2FA via SMS em 2003) e no desenvolvimento de metodologias próprias de Prompt Engineering e Contextos de IA baseados em estruturas de skills e guidelines. Liderou equipes de tecnologia em grandes corporações como TV Globo, L'Oréal e Magazine Luiza, além de ser autor de blog técnico com mais de 320 mil acessos.

*Estrutura Canônica do Manifesto - AI Project Manifest Standard*